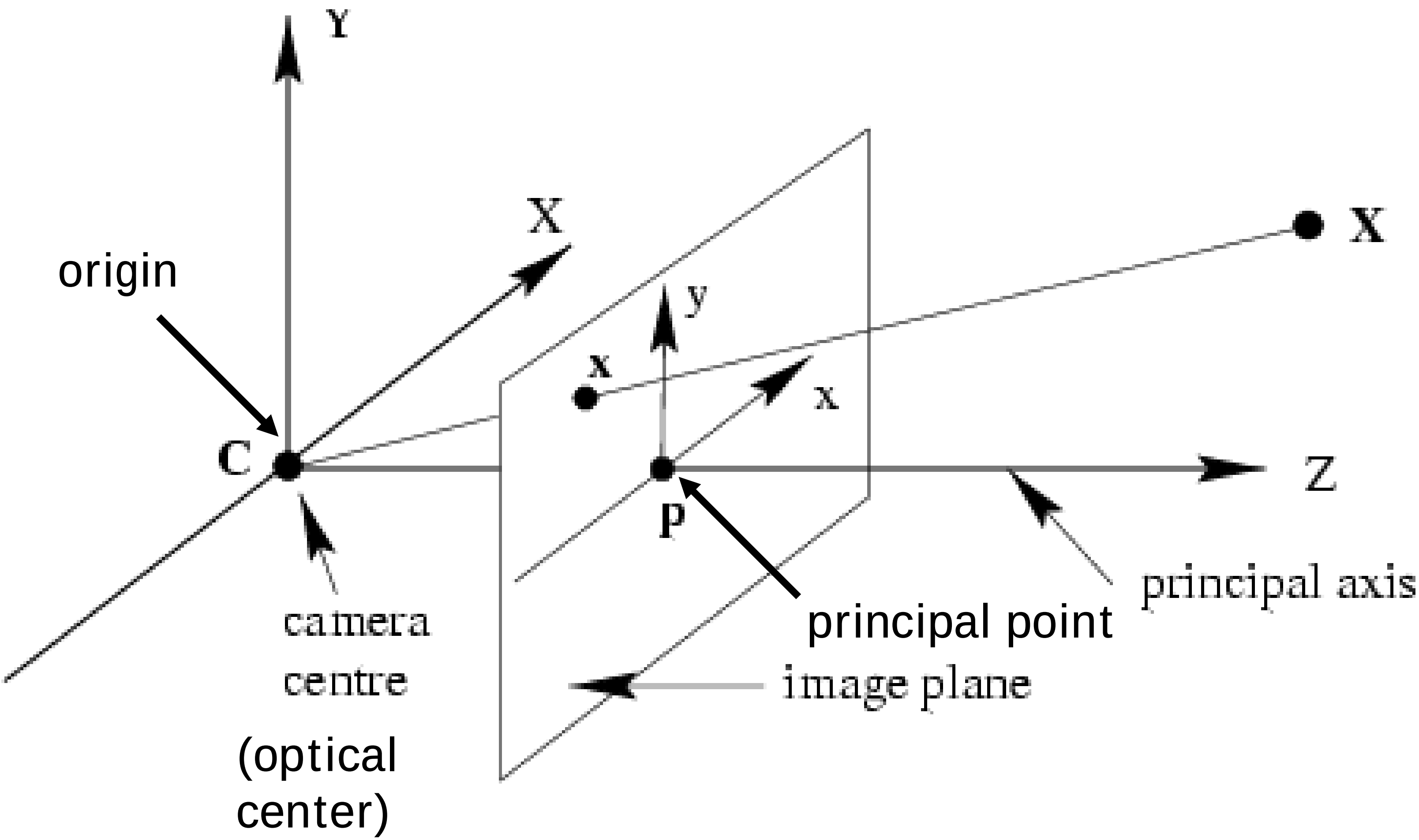
# Computer Vision

## Day 8

2017, Tanta University
**Eng.**Sara Hussien

# Camera Calibration

# Pinhole camera model



# Pinhole camera model

- For finite projective cameras, the correspondence between points in the world and points in the image can be described by the simple model

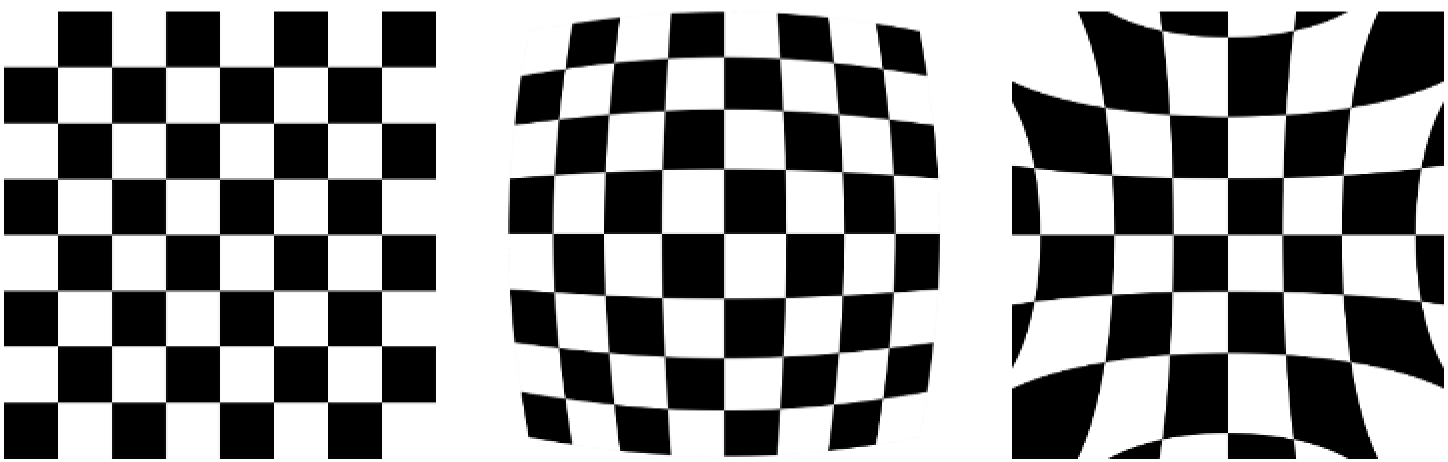
$$P = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} [R \quad \boldsymbol{t}]$$

- This camera model is typically not good enough for accurate geometrical computations based on images it comes ***significant distortion***.
- Luckily, these are ***constants*** and with a ***calibration and some remapping*** we can correct this.
- Furthermore, with calibration you may also determine the relation between the camera's natural units (pixels) and the real world units (for example millimeters).

# Camera Calibration

- For the distortion OpenCV takes into account the radial and tangential factors.
  - For the radial factor one uses the following formula:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$



No distortion

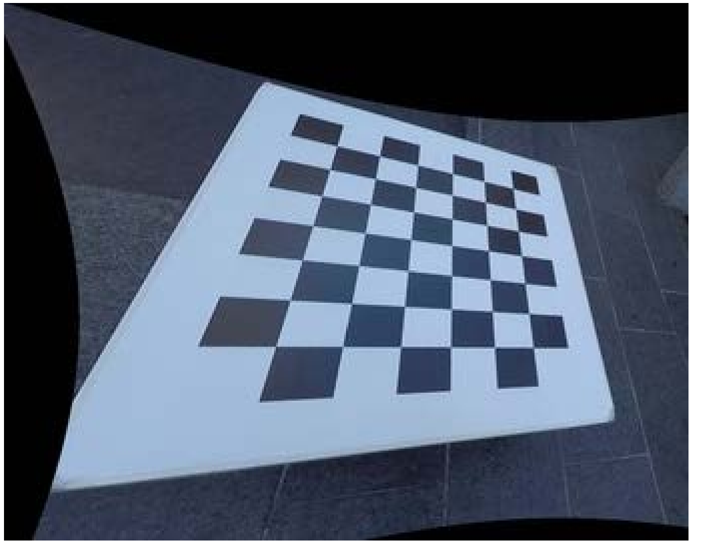
Positive radial distortion (Barrel distortion)

Negative radial distortion (Pincushion distortion)

# Camera Calibration

- For the distortion OpenCV takes into account the radial and tangential factors.
  - **Tangential distortion** occurs because the image taking lenses are not perfectly parallel to the imaging plane. It can be corrected via the formulas:

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$



# Camera Calibration

- So we have five distortion parameters which in OpenCV are presented as one row matrix with 5 columns:

$$\text{Distortion}_{\text{coefficients}} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

- Also, we need to find ***intrinsic*** and **extrinsic** parameters of a camera.
  - Intrinsic parameters : specific to a camera.
    - includes information like **focal length** (fx , fy), **optical centers** (cx , cy).
    - called camera matrix.
    - depends on the camera only
  - Extrinsic parameters corresponds to
    - **rotation** Vectors
    - **translation vectors** which translates a coordinates of a 3D point to a coordinate system.

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

# Camera Calibration

- ***The process of determining these two matrices is the calibration***
- Currently OpenCV supports three types of objects for calibration:
  1. Classical black-white chessboard
  2. Symmetrical circle pattern
  3. Asymmetrical circle pattern

# Camera Calibration

- For stereo applications, these distortions need to be corrected first.
- To find all these parameters
  1. provide some sample images of a well defined pattern (eg, chess board).
  2. We find some specific points in it ( square corners in chess board).
  3. We know its coordinates in real world space and we know its coordinates in image.
  - With these data, some mathematical problem is solved in background to get the distortion coefficients.
  - For better results, we need at least 10 test patterns.

# Camera Calibration

1. find the corners

**Python:** `cv2.findChessboardCorners`(image, patternSize[, corners[, flags]]) → retval, corners

- **image** – Source chessboard view. It must be an 8-bit grayscale or color image.
- **patternSize** – Number of inner corners per a chessboard row and column ( `patternSize = cvSize(points_per_`
  `cvSize(columns,rows)` ).
- **corners** – Output array of detected corners.
- **flags** –

2. increase their accuracy using **cv2.cornerSubPix()**
3. draw the pattern using **cv2.drawChessboardCorners()**
4. we have our object points and image points we are ready to go for calibration **cv2.calibrateCamera()**
5. Undistortion using **cv2.undistort() or remap()**